USB Device Printer Class Driver User Guide

Version 1.00

For use with USBD Printer Class Driver versions 2.03 and above

Exported on 02/01/2019

Table of Contents

1 System Overview

This chapter contains the fundamental information for this module.

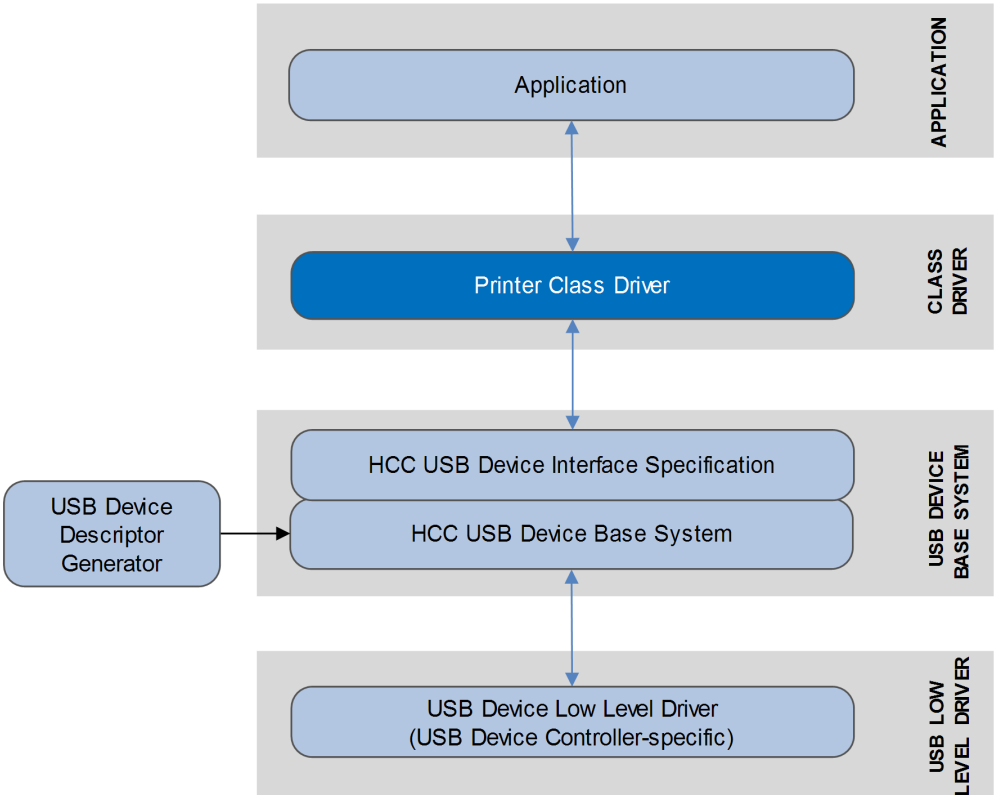
The component sections are as follows:

- Introduction – describes the main elements of the module.
- Feature Check – summarizes the main features of the module as bullet points.
- Packages and Documents – the *Packages* section lists the packages that you need in order to use this module. The *Documents* section lists the relevant user guides.
- Change History – lists the earlier versions of this manual, giving the software version that each manual describes.

1.1 Introduction

This guide is for those who want to implement an embedded USB printer class driver. The printer class driver is used for connecting printers over a USB link to a host system. The **usbd_cd_printer** package is a function device implementation of this class.

The system structure is shown in the diagram below:



On the embedded device side, the printer can be either a real device or a virtual device. This class driver is effectively a library. It provides a set of function calls for an application to use to send and receive data through the serial port. The **usbdprn_init()** function registers the class driver with the Embedded USB Device (EUSBD) base system and this call sets up callbacks for the base system to use.

Note:

- This module is part of HCC's EUSBD system, as described in the *HCC Embedded USB Device Base System User Guide*. This module communicates with the EUSBD base system through the EUSBD Device Interface, as described in the above manual.
- This module conforms to the *Universal Serial Bus Class Definition for Communication Devices Version 1.1*. You can download this recommended reference from www.usb.org.

1.2 Feature Check

The main features of the class driver are the following:

- Conforms to the HCC Advanced Embedded Framework.
- Designed for integration with both RTOS and non-RTOS based systems.
- Supports all devices that conform to the USB printer specification.
- Compatible with sample device files produced by using the *HCC USB Device Descriptor Generator*.
- Uses a system of user-defined callbacks for state change events.

1.3 Packages and Documents

Packages

This table lists the packages that you need in order to use this module:

Package	Description
hcc_base_doc	This contains the two guides that will help you get started.
usbd_base	The USB device base package. This is the framework used by USB class drivers to communicate over USB using a specific USB device controller package.
usbd_cd_printer	The USB device printer class driver package described by this document.
util_hcc_mem	The HCC memory management utility.

Documents

For an overview of HCC's embedded USB stacks, see Product Information on the main HCC website.

Readers should note the points in the HCC Documentation Guidelines on the HCC documentation website.

HCC Firmware Quick Start Guide

This document describes how to install packages provided by HCC in the target development environment. Also follow the *Quick Start Guide* when HCC provides package updates.

HCC Source Tree Guide

This document describes the HCC source tree. It gives an overview of the system to make clear the logic behind its organization.

HCC Embedded USB Device Base System User Guide

This document defines the USB device base system upon which the complete USB stack is built.

HCC USB Device Printer Class Driver User Guide

This is this document.

HCC USB Device Descriptor Generator User Guide

This document describes the tool that creates USB descriptor files for inclusion in a project that uses the EUSBD stack.

1.4 Change History

This section describes past changes to this manual.

- To view or download manuals, see USB Device PDFs.
- For the history of changes made to the package code itself, see History: usbd_cd_printer.

The current version of this manual is 1.00. The full list of versions is as follows:

Manual version	Date	Software version	Reason for change
1.00	2019-02-01	2.03	First online version.

2 Source File List

This section describes all the source code files included in the system. These files follow the HCC Embedded standard source tree system, described in the *HCC Source Tree Guide*. All references to file pathnames refer to locations within this standard source tree, not within the package you initially receive.

> **Note:** Do not modify any files except the configuration file.

2.1 API Header File

The file **src/api/api_usbd_prn.h** is the only file that should be included by an application using this module. For details of the API functions, see Application Programming Interface.

2.2 Configuration File

The file **src/config/config_usbd_prn.h** contains all the configurable parameters of the system. Configure these as required. For details of these options, see Configuration Options.

2.3 Source Code Files

These files are in the directory **src/usb-device/class-drivers/printer**. **These files should only be modified by HCC**.

File	Description
bulk_rx.c	Bulk endpoint handler module source code.
bulk_rx.h	Bulk endpoint handler module header file.
usbd_prn.c	Class driver source code.

2.4 Version File

The file **src/version/ver_usbd_prn.h** contains the version number of this module. This version number is checked by all modules that use this module to ensure system consistency over upgrades.

3 Configuration Options

Set the system configuration options in the file **src/config/config_usbd_prn.h**. This section lists the available configuration options and their default values.

USBDPRN_DEVICE_ID_STRING

A macro defining the IEEE 1284 identification string of the printer device. The default is:

```
"\x28\0MFG:HCC-Embedded Kft;CMD:PCL;MODEL:USB printer demo"
```

USBDPRN_USE_SAFE_RECEIVE

Keep this at the default of 1 to use safe reception in which the class driver performs reception packet by packet. Windows XP and Windows 7 (SP1) need this parameter set to 1.

Enable this if the host is not terminating out transfers with zero length transactions if the length of the transaction is the exact multiple of the maximum packet size of the Bulk endpoint.

USBDPRN_DEVICE_RX_BUFFER_SIZE

The size of the RX buffer allocated by the class driver. If USBDPRN_USE_SAFE_RECEIVE is set to a non-zero value, set this option to match the packet size of the Bulk OUT endpoint. The default is 64.

4 Application Programming Interface

This section documents the Application Programming Interface (API). It includes all the functions that are available to an application program.

4.1 Module Management

The functions are the following:

Function	Description
usbdprn_init()	Initializes the module and allocates the required resources.
usbdprn_start()	Starts the module.
usbdprn_stop()	Stops the module.
usbdprn_delete()	Deletes the module and releases the resources it used.

usbdprn_init

Use this function to initialize the class driver and allocate the required resources.

> **Note:** You must call this before any other function.

Format

```
int usbdprn_init ( void )
```

Arguments

Parameter
None.

Return Values

Return value	Description
USBD_PRN_SUCCESS	Successful execution.
USBD_PRN_ERROR	Operation failed.

usbdprn_start

Use this function to start the module.

> **Note:** You must call **usbdprn_init()** before this to initialize the module.

Format

```
int usbdprn_start ( void )
```

Arguments

Parameter
None.

Return Values

Return value	Description
USBD_PRN_SUCCESS	Successful execution.

usbdprn_stop

Use this function to stop the module.

Format

```
int usbdprn_stop ( void )
```

Arguments

Parameter
None.

Return Values

Return value	Description
USBD_PRN_SUCCESS	Successful execution.

usbdprn_delete

Use this function to delete the module and release the associated resources.

Format

```
int usbdprn_delete ( void )
```

Arguments

Parameter
None.

Return Values

Return value	Description
USBD_PRN_SUCCESS	Successful execution.

4.2 Printer Management

The functions are the following:

Function	Description
usbd_prn_get_port_state()	Reads the current port state bitmask.
usbd_prn_set_port_state()	Sets the port state bitmask.
usbdprn_get_rx_num()	Gets the number of bytes available in the RX buffer.
usbdprn_is_active()	Checks whether the USB connection to the host is alive.
usbdprn_receive()	Receives data from the host.
usbdprn_set_rst_cb()	Registers a callback function and its parameter value.
usbdprn_set_start_stop_cb()	Specifies a **start_stop_cb**() function and its parameter value.

usbd_prn_get_port_state

Use this function to read the current port state bitmask.

Format

```
int usbd_prn_get_port_state ( uint8_t * state_mask )
```

Arguments

Parameter	Description	Type
state_mask	The port state bitmask.	uint8_t *

Return Values

Return value	Description
USBD_PRN_SUCCESS	Successful execution.
USBD_PRN_BUSY	The printer is busy.
USBD_PRN_ERROR	Operation failed.

usbd_prn_set_port_state

Use this function to set the port state bitmask.

The bitmask is returned to the host when it reads the value.

Format

```
int usbd_prn_set_port_state ( uint8_t state_mask )
```

Arguments

Parameter	Description	Type
state_mask	The port state bitmask.	uint8_t

Return Values

Return value	Description
USBD_PRN_SUCCESS	Successful execution.
USBD_PRN_ERROR	Operation failed.

usbdprn_get_rx_num

Use this function to get the number of bytes available in the RX buffer.

Format

```
uint32_t usbdprn_get_rx_num ( void )
```

Arguments

Parameter
None.

Return Values

Return value
The number of unread bytes in the buffer.

usbdprn_is_active

Use this function to check whether the USB connection to the host is alive.

Format

```
int usbdprn_is_active ( void )
```

Arguments

Parameter
None.

Return Values

Return value	Description
0	USB is not ready to communicate.
1	USB is operating.

usbdprn_receive

Use this function to receive data from the host.

If an RTOS is present, this call is blocking.

Format

```
int usbdprn_receive (
    void *       buffer,
    uint32_t     length,
    uint32_t *   read )
```

Arguments

Parameter	Description	Type
buffer	Where to store the received data.	void *
length	The size of the buffer.	uint32_t
read	On return, the actual number of bytes read. This can be less than *length*.	uint32_t *

Return Values

Return value	Description
USBD_PRN_SUCCESS	Successful execution.
USBD_PRN_BUSY	The printer is busy.
USBD_PRN_ERROR	Operation failed.

usbdprn_set_rst_cb

Use this function to register a callback function and its parameter value.

The notification function **rst_cb** is called by the USB stack when a reset printer command is received on the control channel.

> **Note:** It is the user's responsibility to provide any callback functions the application requires. Providing such functions is optional.

Format

```
void  usbdprn_set_rst_cb (
    void ( *   fn )( uint32_t param ),
    uint32_t   param )
```

Arguments

Parameter	Description	Type
fn	A pointer to the callback function.	void *
param	The value of the parameter to pass to the callback function.	uint32_t

Return Values

Return value	Description
USBD_PRN_SUCCESS	Successful execution.
USBD_PRN_ERROR	Operation failed.

usbdprn_set_start_stop_cb

Use this function to specify a **start_stop_cb**() function and its parameter value.

The notification function **start_stop_cb()** is called by the USB stack when the class driver is started or stopped. You can call **usbdprn_is_active()** from the callback to see whether a start or a stop triggered it.

Note: It is the user's responsibility to provide any callback functions the application requires. Providing such functions is optional.

Format

```
void  usbdprn_set_start_stop_cb (
    void ( *   fn )( uint32_t param ),
    uint32_t   param )
```

Arguments

Parameter	Description	Type
fn	A pointer to the callback function.	void *
param	The value of the parameter to pass to the callback function.	uint32_t

Return Values

Return value	Description
USBD_PRN_SUCCESS	Successful execution.

4.3 Error Codes

If a function executes successfully it returns with USBD_PRN_SUCCESS, a value of 0. The following table shows the meaning of the error codes.

Return Value	Value	Description
USBD_PRN_SUCCESS	0	Successful execution.
USBD_PRN_ERROR	1	Operation failed.
USBD_PRN_BUSY	2	Driver is busy.

4.4 Types and Definitions

The single section here describes the port state bitmasks that are defined in the API Header file.

Port State Bitmasks

The port state bitmasks are as follows:

Return Value	Value	Description
USBDPRN_PST_PEPEER_EMPTY	1U << 5	The device has no paper.
USBDPRN_PST_SELECT	1U << 4	The device is selected.
USBDPRN_PST_NOT_ERROR	1U << 3	The device encountered no error.

Note: These are unique bits in a set of bit values.

5 Integration

This section specifies the elements of this package that need porting, depending on the target environment.

5.1 OS Abstraction Layer

All HCC modules use the OS Abstraction Layer (OAL) that allows the module to run seamlessly with a wide variety of RTOSes, or without an RTOS.

This module uses the following OAL components:

Resource	Requirement
Tasks	0
Mutexes	0
Events	1

5.2 Utilities

This module uses the **util/mem/hcc_mem.h** utility.

5.3 PSP Porting

The Platform Support Package (PSP) is designed to hold all platform-specific functionality, either because it relies on specific features of a target system, or because this provides the most efficient or flexible solution for the developer.

This module makes use of the following standard PSP function:

Function	Package	Component	Description
psp_memcpy()	psp_base	psp_string	Copies a block of memory. The result is a binary copy of the data.